Konzept zur Einführung eines Software-Modells für die Berechnung einer Verkehrsroute mit möglichst geringen CO2-Emissionen.

Autoren: Mario Bolte (Kurslehrer), Kira Lomholt-Busch, Nicolas Constantin Moch, Maike Foyer, Nils Schumacher, Levin Gehse, Alexander Namokel, Finja Conrad, Adrian Lopez-Ellermann, Emma Prawiro, Lutz Pelster.
Informatik Q1-Grundkurs des Evangelisch Stiftischen Gymnasiums in Gütersloh

1 Einleitung

In Verbindung mit der angekündigten Einführung der Öko-Route auf Google Maps haben wir einen genaueren Blick auf die Funktionsweise geworfen, indem wir eine einfache Routennavigation implementiert und anschließend ein Modell für eine Öko-Route entwickelt haben.
Uns ist Umweltschutz sehr wichtig und wir finden, dass er für jeden verfügbar und im Alltag inkorporiert sein sollte. Wir denken, dass eine Öko-Route ein vielversprechender Schritt sein könnte und haben uns näher damit auseinandergesetzt. Im Unterricht haben wir uns die Frage gestellt, wie sinnvoll diese Funktion ist und ein Konzept für eine mögliche Umsetzung erarbeitet. Die Ergebnisse möchten wir Ihnen nun präsentieren.

2 Grundlagen

Bevor wir unser Konzept zur Umsetzung einer Ökoroute erläutern, möchten wir zunächst auf die beiden zentralen Bestandteile Routennavigation und CO2-Ausstoß im Straßenverkehr eingehen.

2.1 Routennavigation

Dem Problem der Routennavigation liegt die Graphentheorie zugrunde. Ein Graph definiert sich über eine Menge von Knoten und über eine Menge von Kanten [G = (V , E)]. Für Graphen gibt es verschiedene Algorithmen zur Berechnung einer Route, also einer Verbindung von einem Knoten zu einem anderen Knoten. Der Dijkstra-Algorithmus ist ein solcher Algorithmus, der zur Lösung von Optimierungsproblemen, also zum Beispiel zur Berechnung der kostengünstigsten beziehungsweise kürzesten Strecke, verwendet wird. Die Kosten werden dabei mithilfe von Knoten und Kanten berechnet. In einem Navigationssystem zur Wegberechnung und Optimierung wäre ein Knoten zum Beispiel ein bestimmter Standort, also ein Ausgangspunkt, ein Wegpunkt, eine Kreuzung oder ein Ziel in unserem Datenmodell. Eine Kante beschreibt im Navigationssystem eine Straße, in unserem Datenmodell zum Beispiel die “Koenigsstrasse”. Eine Kante verbindet immer zwei Knoten aus dem Graphen.
Die Abbildung 1 zeigt das Verkehrsnetz rund um unsere Schule inklusive angrenzendem Sportplatz. Mit Hilfe des Algorithmus haben wir eine Routennavigation implementiert, die die kürzeste Route mit rot markierten Knoten visualisiert.

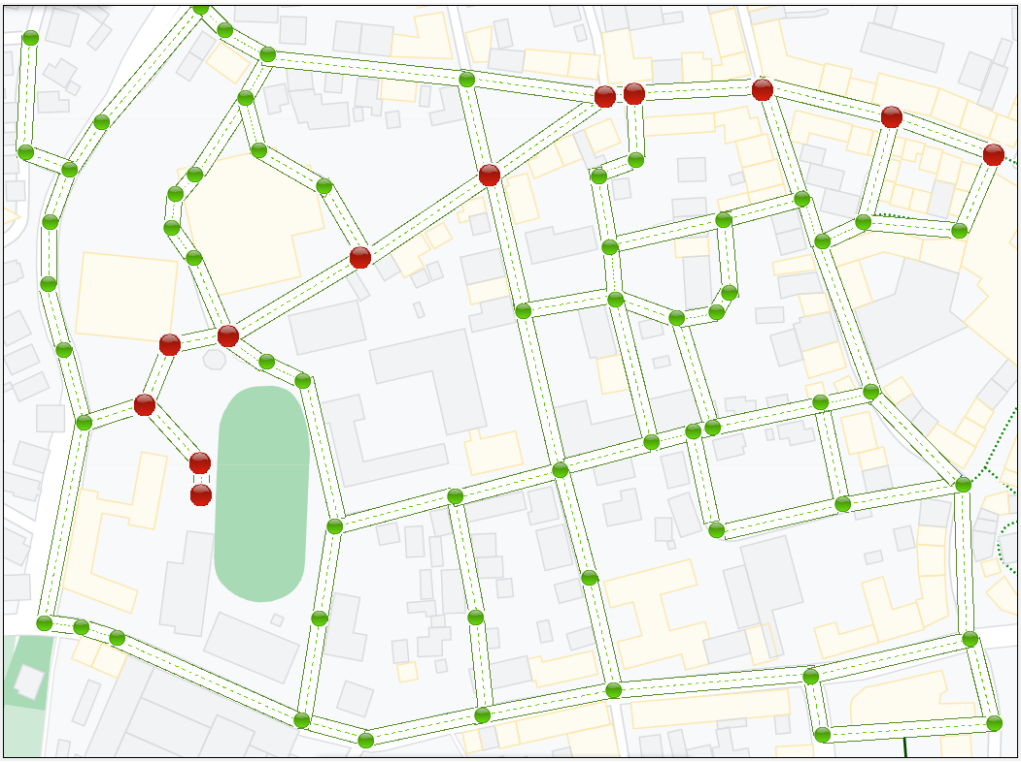


Abb. 1: Verkehrsroute am ESG in Gütersloh

Der Dijkstra-Algorithmus ist ein Greedy-Algorithmus, der vom Startknoten ausgehend immer die geringsten Kosten berechnet, um zu einem anderen Knoten zu gelangen. Die Funktion des Algorithmus ist im folgenden als Pseudocode dargestellt (Vgl. https://de.wikipedia.org/wiki/Dijkstra-Algorithmus , Zugriff am 06.05.2022).

FUNKTION DES ALGORITHMUS

1. **Initialisierung:** Startknoten hat Kosten=0, alle anderen ∞
2. Knoten mit **geringsten Kosten** wird ausgewählt → „aktuell besucht“ (Knoten werden nicht beachtet)

Update:

3. **Addiere** für jeden Nachbarsknoten einzeln: Kosten von „aktuell besucht“ + Kosten bis zum Nachbarsknoten (Knoten werden nicht beachtet)
4. **Neu berechnete Kosten geringer** als bisher gespeicherte Kosten des Nachbarsknoten?

   → durch neu berechnete **Kosten ersetzen + Vorgängerknoten abspeichern**

   + Knoten kommt in Liste der bisher **besuchten** Knoten
5. **Beenden,** wenn alle **„besucht“** wurden

2.1 CO2-Ausstoß im Straßenverkehr

Um die CO2-Emissionen eines PKWs zu berechnen, müssen unterschiedliche Faktoren beachtet werden. Zunächst ist zu erwähnen, dass der CO2-Ausstoß an den Kraftstoffverbrauch gekoppelt ist. Wenn sich der Verbrauch erhöht, erhöht sich ebenfalls der CO2-Ausstoß. Der Verbrauch des Kraftstoffes eines Fahrzeugs hängt von einer Vielzahl von Kriterien ab.

Tabelle: CO_2-Emissionen von Autos: CO_2- Äquivalent in g/l bei Benzin und Diesel

Kraftstoffe	CO_2 in Gramm pro Liter
Benzin	2.370
Diesel	2.650

[Q1]

Zuerst wird auf Eigenschaften des Fahrzeuges eingegangen. Es wird in Fahrzeuge unterteilt, welche entweder Diesel oder Benzin als Kraftstoff nutzen. Fahrzeuge, welche Diesel als Kraftstoff verwenden, haben auf den Liter gesehen einen höheren CO2-Ausstoß, verbrauchen aber im Schnitt weniger Treibstoff, sodass insgesamt bei dem Diesel ein minimaler Vorteil besteht.

Als zweite Unterteilung für das Modell kann man Fahrzeuge in die Kategorien Kleinwagen, Mittelklasse und Oberklasse unterteilen. Dies dient als Vereinfachung, da es sonst zu viele einzelne Faktoren zu berücksichtigen gäbe. Mit einem größeren Fahrzeug geht gleichzeitig ein höherer Luftwiderstand und ein höheres Gewicht einher. Der letzte Faktor, der für das Modell berücksichtigt wird, ist die Straße, auf der gefahren wird. Diese wird in Stadtverkehr, Landstraße und Autobahn unterteilt. In der Stadt ist der Verbrauch am größten, da hier sehr oft in kurzer Zeit beschleunigt und gebremst wird, was den Kraftstoffverbrauch stark ansteigen lässt. Auf der Landstraße ist der Verbrauch am geringsten, da hier über längere Strecken zwar eine höhere Geschwindigkeit gefahren wird, aber hier nicht so häufig verzögert oder beschleunigt wird. Die Autobahn ordnet sich vom Kraftstoffverbrauch zwischen den andern beiden ein, da hier die Geschwindigkeit viel höher ist, was den Kraftstoffverbrauch erhöht. Bremsvorgänge und Beschleunigungen kommen allerdings nicht häufig vor.

Durchschnittlicher Ausstoß eines Autos: CO_2 in kg pro 100 km (Kraftstoffverbrauch in Liter)

	Stadtverkehr	Landstraße	Autobahn
Otto-Pkw (Benziner)			
Kleinwagen	18 (7,3)	13 (5,1)	17 (6,9)
Mittelklasse	25 (8,7)	14 (5,7)	19 (7,4)
Oberklasse	32 (12,6)	22 (8,7)	28 (11,1)
Diesel-Pkw			
Kleinwagen	14 (4,5)	10 (3,1)	13 (4,1)
Mittelklasse	21 (6,8)	15 (4,7)	16 (5,2)
Oberklasse	28 (8,9)	19 (6,2)	26 (8,5)

[Q2]

Faktoren, die nicht in das Modell mit einbezogen werden, haben entweder nur einen geringfügigen Einfluss oder sind nicht oder nur umständlich erfassbar. Dazu gehört das Alter des Autos, da sich die Effizienz der Motoren in den letzten Jahrzehnten zwar verbessert, aber nur geringfügig verändert hat. Der nächste

Faktor ist der Reifendruck, welcher im Optimalzustand eine Differenz von fünf Prozent erwirkt (Vgl. [Q3]).
Als Nächstes sind die menschlichen Faktoren zu nennen. Diese sind beispielsweise die Fahrweise und die Tatsache, ob man die Klimaanlage verwendet. Da für die Klimaanlage ein weiterer Generator notwendig ist, welcher Energie vom Motor verbraucht, wird dadurch die Leistung reduziert und der Kraftstoffverbrauch erhöht. Dies kann unser Modell jedoch nicht mit einbeziehen, da unsere Entwicklungsumgebung keine direkte Verbindung zum Fahrzeug hat. Die Fahrweise hat einen noch viel größeren Einfluss von bis zu 30 Prozent auf den Spritverbrauch, weshalb durch starke Beschleunigungs- und Bremsmanöver viel Energie verbraucht wird (Vgl. [Q3]). Dies hängt damit zusammen, dass die Beschleunigung sehr kraftaufwendig ist im Gegensatz zur konstant eingehaltenen Geschwindigkeit.
Weiterhin gibt es noch mehr Faktoren, die berücksichtigt werden könnten, diese haben aber nur einen geringfügigen Einfluss auf den CO2-Ausstoß.

3 Technisches Datenmodell

Für die Umsetzung einer einfachen Routennavigation mit dem Ziel, die kürzeste Verbindung zu finden, nutzen wir die objektorientierte Programmiersprache Java mit Greenfoot als Entwicklungsumgebung. Daher beruht unser theoretisches Modell ebenfalls auf diesem Ansatz.

Zunächst muss das Modell die CO2-Werte kennen, um eine möglichst emissionsarme Route zu berechnen. Diese hängen, wie zuvor erläutert, von verschiedenen Faktoren ab. Dazu wird in jeder Straße (Kante) für jedes Automodell jeweils ein konkreter CO2-Wert gespeichert. Hier unterscheiden wir zwischen Diesel- und Benzinmotoren und jeweils den Autokategorien: Kleinwagen, Mittel- und Oberklasse, sodass es insgesamt sechs mögliche Werte gibt.
Nachdem der User angegeben hat, welches Automodell er besitzt, kann mittels einer Formel der passende CO2-Wert für jede Straße berechnet werden. Dies erfolgt durch die drei Parameter: Automodell, Distanz der Straße und ob die Straße sich innerorts oder außerorts befindet oder eine Autobahn ist. Daraus ergibt sich Folgendes:

CO2-Kosten = Durchschnittlicher Ausstoß in kg/100km (siehe Q2) * 100 * Distanz (m) einer Kante (g)

In unserem vereinfachten Modell sind die CO2-Werte statisch, also gleichbleibende Durchschnittswerte. Unternehmen mit größerer Datenmacht können dies optimiert durchführen, indem sie die Werte individuell und variabel berechnen. Hierdurch können sich ändernde Faktoren, wie zum Beispiel die Verkehrslage miteinbezogen werden und es muss nicht auf Durchschnittswerte zurückgegriffen werden.

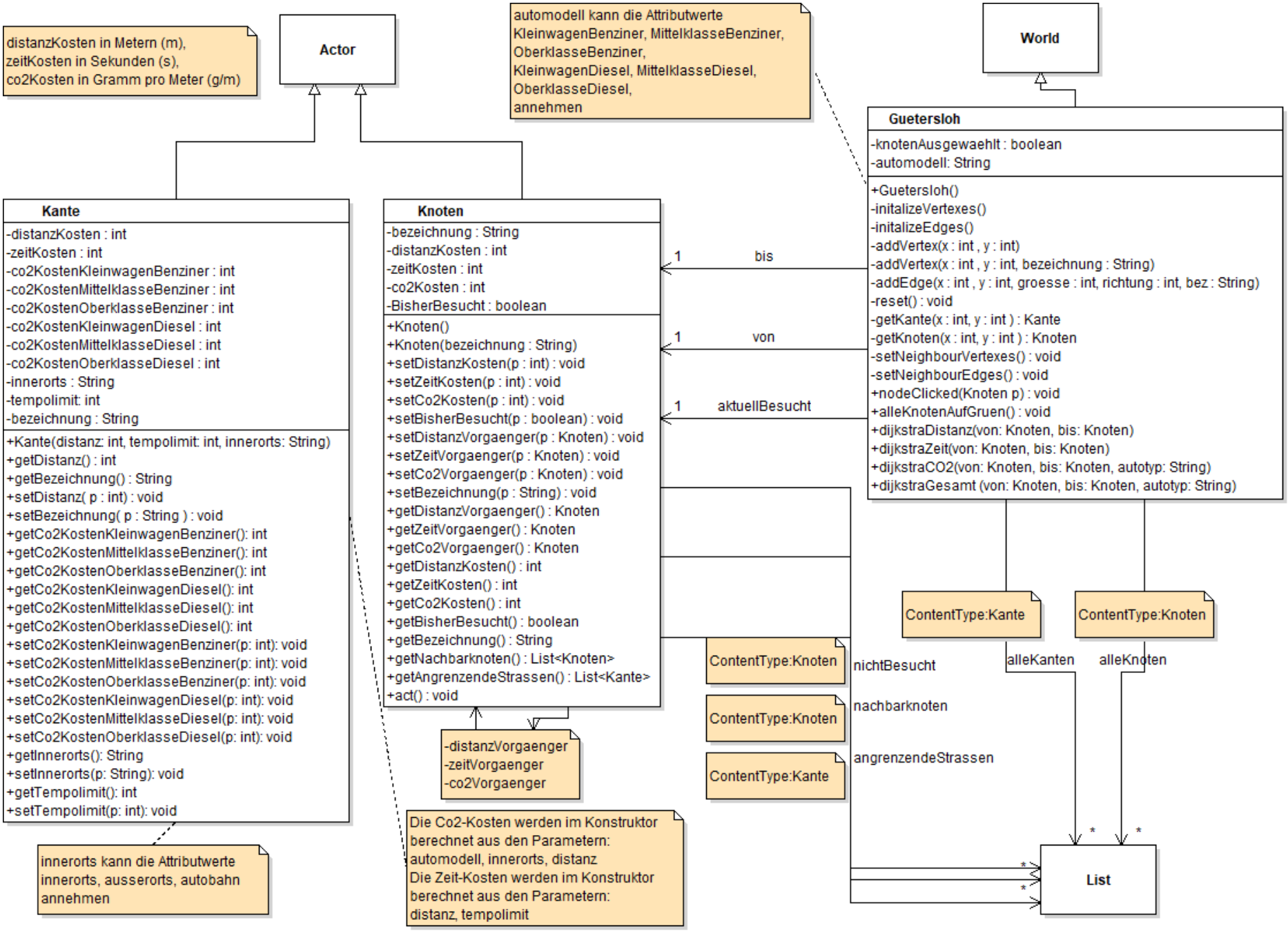
Die Berechnung der emissionsärmsten Route erfolgt auch hier durch den Dijktstra-Algorithmus. Dieser berechnet normalerweise die kürzeste Route von A nach B, verwendet hier aber anstelle von Distanz-Kosten, die CO2-Kosten.
Im Folgenden wird das Implementationsdiagramm unseres Modells gezeigt. Die Klasse Gütersloh hat drei verschiedene Dijkstra-Algorithmen, die vom User ausgewählt werden können. DijkstraDistanz berechnet die kürzeste Route, dijkstraZeit die zeitlich schnellste Route und dijkstraCO2 die emmisionsärmste Route, welche zuvor erläutert wurde.

DijkstraGesamt zeigt dem User alle drei möglichen Routen an, sowie deren Distanz, benötigte Zeit und CO2-Kosten.

Das folgende Implementationsdiagramm (siehe S.6) stellt den Bauplan unseres Algorithmus dar. Es unterteilt sich in die drei Klassen Knoten und Kante, welche beide Unterklassen von Actor sind, sowie der Klasse Gütersloh, die eine Unterklasse von World ist. Außerdem wird die Klasse List verwendet.
Die Klasse Gütersloh stellt die verwaltende Klasse dar. Sie besitzt die Variable automodell und Methoden zur Verwaltung bzw. Erstellung von Knoten und Kanten. Zusätzlich kann der User über diese Klasse die Dijkstra-Methoden durchführen, für die er den Start- und Zielort (Knoten von und Knoten bis) und gegebenenfalls das Automodell angeben muss. Die Klasse Gütersloh besitzt jeweils eine Liste, in der alle Knoten und eine Liste, in der alle Kanten gespeichert sind. Darüber hinaus besitzt sie eine Objektreferenz zum Knoten bis (Zielort), dem Knoten von (Startort) und dem Knoten aktuellBesucht (Knoten, der zurzeit geprüft wird).

Diese gehören der Klasse Knoten an, deren Objekte die möglichen Orte darstellen und immer eine Bezeichnung, also den Standortnamen, gespeichert haben. Zur Durchführung des Dijkstra-Algorithmus benötigen sie des Weiteren eine Variable Kosten, in der die bisher errechneten geringsten Kosten gespeichert werden. In der Variable Vorgängerknoten ist eine Objektreferenz zum Vorgängerknoten abgespeichert, der später den kürzesten Weg zu diesem Knoten angeben wird. Sowohl die Kosten, als auch die Vorgängerknoten müssen für die verschiedenen Dijkstra-Methoden eine eigene Variable haben, da die Kosten und die daraus resultierende Route für Distanz, Zeit und CO2-Bilanz unterschiedlich sind.
Außerdem besitzt jeder Knoten eine Objektreferenz zu einer Liste, in der alle seine Nachbarknoten gespeichert sind, sowie eine Liste, die seine angrenzenden Straßen speichert. Zudem speichert die Liste nichtBesucht alle Knoten ab, für die vom Algorithmus bisher noch nicht die günstigsten Kosten, also der optimale Weg dorthin berechnet wurden. Auch besitzt jeder Knoten selber eine Variable bisherBesucht, die nach der Überprüfung auf true gesetzt wird.
Die letzte Klasse Kante stellt die Straßen in unserem Modell dar. In den Kosten-Variablen speichert sie ab, welche Kosten aufgebracht werden müssen, um die Straße zu verwenden. DistanzKosten sagt beispielsweise aus, dass die Straße 500m lang ist, was folglich zu der Route hinzuaddiert werden muss. Im Konstruktor der Klasse Kante werden jeweils die CO2-Kosten für die Straßen aus den Parametern Automodell, Innerorts und Distanz berechnet und in der zugehörigen Variable gespeichert. Des Weiteren werden die Zeit-Kosten für die jeweilige Straße, auch im Konstruktor der Klasse Kante, aus den Parametern Distanz und Tempolimit errechnet.

Trotz der großen Menge an Variablen und Methoden im Modell ist die Komplexität des Programms kaum größer als die eines regulären Dijkstra-Algorithmus. Es deckt lediglich mehr Sonderfälle ab, sodass der User eine Route nach seinen gewünschten Kategorien berechnen kann.


distanzKosten in Metern (m),
zeitKosten in Sekunden (s),
co2Kosten in Gramm pro Meter (g/m)
Actor
automodell kann die Attributwerte
KleinwagenBenziner, MittelklasseBenziner,
OberklasseBenziner,
KleinwagenDiesel, MittelklasseDiesel,
OberklasseDiesel,
annehmen
World
Guetersloh
-knotenAusgewaehlt : boolean
-automodell: String
+Guetersloh()
-initalizeVertexes()
-initalizeEdges()
-addVertex(x : int , y : int)
-addVertex(x : int , y : int, bezeichnung : String)
-addEdge(x : int , y : int, groesse : int, richtung : int, bez : String)
-reset() : void
-getKante(x : int, y : int) : Kante
-getKnoten(x : int, y : int) : Knoten
-setNeighbourVertexes() : void
-setNeighbourEdges() : void
+nodeClicked(Knoten p) : void
+alleKnotenAufGruen() : void
+dijkstraDistanz(von: Knoten, bis: Knoten)
+dijkstraZeit(von: Knoten, bis: Knoten)
+dijkstraCO2(von: Knoten, bis: Knoten, autotyp: String)
+dijkstraGesamt (von: Knoten, bis: Knoten, autotyp: String)
Kante
-distanzKosten : int
-zeitKosten : int
-co2KostenKleinwagenBenziner : int
-co2KostenMittelklasseBenziner : int
-co2KostenOberklasseBenziner : int
-co2KostenKleinwagenDiesel : int
-co2KostenMittelklasseDiesel : int
-co2KostenOberklasseDiesel : int
-innerorts : String
-tempolimit: int
-bezeichnung : String
+Kante(distanz: int, tempolimit: int, innerorts: String)
+getDistanz() : int
+getBezeichnung() : String
+setDistanz(p : int) : void
+setBezeichnung(p : String) : void
+getCo2KostenKleinwagenBenziner(): int
+getCo2KostenMittelklasseBenziner(): int
+getCo2KostenOberklasseBenziner(): int
+getCo2KostenKleinwagenDiesel(): int
+getCo2KostenMittelklasseDiesel(): int
+getCo2KostenOberklasseDiesel(): int
+setCo2KostenKleinwagenBenziner(p: int): void
+setCo2KostenMittelklasseBenziner(p: int): void
+setCo2KostenOberklasseBenziner(p: int): void
+setCo2KostenKleinwagenDiesel(p: int): void
+setCo2KostenMittelklasseDiesel(p: int): void
+setCo2KostenOberklasseDiesel(p: int): void
+getInnerorts(): String
+setInnerorts(p: String): void
+getTempolimit(): int
+setTempolimit(p: int): void
Knoten
-bezeichnung : String
-distanzKosten : int
-zeitKosten : int
-co2Kosten : int
-BisherBesucht : boolean
+Knoten()
+Knoten(bezeichnung : String)
+setDistanzKosten(p : int) : void
+setZeitKosten(p : int) : void
+setCo2Kosten(p : int) : void
+setBisherBesucht(p : boolean) : void
+setDistanzVorgaenger(p : Knoten) : void
+setZeitVorgaenger(p : Knoten) : void
+setCo2Vorgaenger(p : Knoten) : void
+setBezeichnung(p : String) : void
+getDistanzVorgaenger() : Knoten
+getZeitVorgaenger() : Knoten
+getCo2Vorgaenger() : Knoten
+getDistanzKosten() : int
+getZeitKosten() : int
+getCo2Kosten() : int
+getBisherBesucht() : boolean
+getBezeichnung() : String
+getNachbarknoten() : List<Knoten>
+getAngrenzendeStrassen() : List<Kante>
+act() : void
1
bis
1
von
1
aktuellBesucht
ContentType:Kante
ContentType:Knoten
alleKanten
alleKnoten
ContentType:Knoten
nichtBesucht
ContentType:Knoten
nachbarknoten
ContentType:Kante
angrenzendeStrassen
-distanzVorgaenger
-zeitVorgaenger
-co2Vorgaenger
*
*
*
*
List
innerorts kann die Attributwerte
innerorts, ausserorts, autobahn
annehmen
Die Co2-Kosten werden im Konstruktor
berechnet aus den Parametern:
automodell, innerorts, distanz
Die Zeit-Kosten werden im Konstruktor
berechnet aus den Parametern:
distanz, tempolimit

4 Fazit

Mit Hilfe unserer Software lassen sich die wichtigsten Einflussfaktoren für den CO2-Ausstoß im Straßenverkehr erfassen, sodass unsere Software die tatsächlichen CO2-Emissionen größtenteils annähern dürfte. Wir gehen davon aus, dass für eine durchschnittliche Verkehrssituation und Fahrweise die Ergebnisse aussagekräftig wären. Durch das steigende Umweltbewusstsein in der Breite der Gesellschaft kann diese Technologie unserer Ansicht nach einen wichtigen Beitrag zur Emissionsreduzierung leisten, daher begrüßen wir sehr, dass Unternehmen diese Option für die Nutzer entwickeln.

4.1 Kritische Reflexion

Aus der Tatsache, dass nur statistische Mittelwerte benutzt werden, folgt, dass unser Datenmodell für Hauptverkehrszeiten vermutlich keine guten Ergebnisse liefern würde. Hier geht das Modell davon aus, dass auf einer Landstraße durchgängig mit wenig Bremsvorgängen gerollt werden kann. Tatsächlich ist zwischen Rheda-Wiedenbrück und Gütersloh zur Hauptverkehrszeit ein Stop-and-Go-Verkehr zu beobachten.
Des Weiteren ist zu erwähnen, dass unser Modell von einem durchschnittlichen Fahrer ausgeht, was aber bei einer aggressiven Fahrweise zu sehr großen Abweichungen führen würde (Vgl. [Q2]).

4.2 Ausblick

Eine wesentliche Verbesserungsmöglichkeit sehen wir über die Routennavigation per Handy in Kombination mit einem sehr großen Datenmodell, welche die aktuelle Verkehrssituation abbildet. Über die anonyme Echtzeitauswertung aller Handys, die eine Verkehrsnavigation im Straßenverkehr nutzen, könnte das Datenmodell permanent aktualisiert werden. So ließen sich im Datenmodell die CO2-Kosten einer Kante anpassen, wenn beispielsweise auf der Landstraße die Durchschnittsgeschwindigkeit von beispielsweise 60 km/h auf 25 km/h fällt. Dies würde dann dazu führen, dass die von der Software errechneten CO2-Emissionen den tatsächlich verursachten Emissionen der aktuellen Verkehrslage deutlich näher kommt. Eine weitere Möglichkeit zur Verbesserung wäre, dass die CO2-Emissionen nicht nur allein aus der aktuellen Verkehrslage berechnet werden, sondern die Fahrweise, welchen einen hohen Einfluss auf die Emissionen hat (Vgl. [Q2]), anhand der Handydaten mit in die Berechnung miteinbezogen wird. Über die Beschleunigung und Geschwindigkeit des Fahrzeugs, die sich anhand der GPS-Koordinaten und Beschleunigungssensoren errechnen lassen, kann die Fahrweise mit in die Berechnung einfließen.

Verbesserung durch:	auf kg CO_2/100 km	Maximaler Unterschied zur ursprünglichen Fahrweise
Niedrigtourige Fahrweise	13,5	minus 20 %
Optimaler Reifendruck	16	minus 5 %
Regelmäßiges Motorwarten	15,5	minus 10 %
Sparsame Fahrweise	13	minus 25 %

[Q2]

Als zusätzliches optionales Feature könnte zur Anregung einer kritischen Reflexion der Fahrweise des Fahrers eine Rückmeldung an den Fahrer erfolgen. Während der Fahrt könnte die Software in regelmäßigen Abständen anzeigen, welche CO2-Emissionen im Idealfall hätten erreicht werden können und welche CO2-Emissionen der Fahrer durch seine tatsächliche Fahrweise erzielt hat. So könnte der Fahrer die eigene Fahrweise mit dem Ziel einer geringeren CO2-Emission anpassen.

Zum Schluss sei noch erwähnt, dass noch zu prüfen wäre, ob die Software für Elektroautos ebenfalls zu einer Reduzierung des Stromverbrauchs genutzt werden kann. Unsere aktuelle Version hilft nur Fahrern von reinen Verbrennungsmotoren.

4 Quellenverzeichnis

[Q1]: https://www.helmholtz.de/newsroom/artikel/wie-viel-co2-steckt-in-einem-liter-benzin/#:~:text=Wenn%20ein%20Fahrzeug%20einen%20Liter,und%20besteht%20aus%20vielen%20Kohlenwasserstoffketten. , Zugriff am 10.05.2022

[Q2]: https://www.co2online.de/klima-schuetzen/mobilitaet/auto-co2-ausstoss/#:~:text=CO2-Ausstoß%20von%20Auto,Kilometer%20(6%20l%20x%202.370) , Zugriff am 06.05.2022

[Q3]: https://www.co2online.de/klima-schuetzen/mobilitaet/auto-co2-ausstoss/#:~:text=CO2-Ausstoß%20von%20Auto,Kilometer%20(6%20l%20x%202.370) , Zugriff am 06.05.2022